Expresiones Lambda

Las expresiones lambda en Java pueden clasificarse según el contexto en el que se utilizan y el tipo de función que representan. Aquí hay una clasificación común:

**UnaryOperator**: Una expresión lambda que toma un solo parámetro y devuelve un resultado del mismo tipo que el parámetro. Por ejemplo:

UnaryOperator<Integer> square = x -> x \* x;

**BinaryOperator**: Similar a UnaryOperator, pero toma dos parámetros del mismo tipo y devuelve un resultado del mismo tipo. Por ejemplo:

BinaryOperator<Integer> sum = (x, y) -> x + y;

**Predicate**: Una expresión lambda que toma un parámetro y devuelve un valor booleano. Usado típicamente para filtrar elementos. Por ejemplo:

Predicate<Integer> isEven = x -> x % 2 == 0;

**Function**: Toma un parámetro y devuelve un resultado. Los tipos de parámetro y resultado pueden ser diferentes. Por ejemplo:

Function<Integer, String> intToString = x -> "Número: " + x;

**BiFunction:** toma dos argumentos y produce un resultado. Es similar a Function, pero acepta dos parámetros en lugar de uno.

BiFunction<String, String, String> concatenate = (s1, s2) -> s1 + s2;

**Consumer**: Toma un parámetro pero no devuelve ningún resultado. Usado típicamente para realizar acciones sobre elementos. Por ejemplo:

Consumer<String> printer = s -> System.out.println(s);

**Supplier**: No toma ningún parámetro pero devuelve un resultado. Usado típicamente para generar valores. Por ejemplo:

Supplier<Double> randomNumber = () -> Math.random();